Deterministic Convergence of Neural Tangent Kernels in Artificial Neural Networks

Harpreet Singh Sumal, Quinton Goos

April 2020

Abstract

Using the knowledge of how kernels are used in Gaussian Processes to estimate features of the data-model, we can combine that with the fact that Artificial Neural Networks (ANNs) are equivalent to Gaussian Processes (GP) at initialization. This allows us to explore kernel methods with ANNs, specifically convergence of kernel methods that can help future studies map an algorithm that takes ANNs with infinite-width limits and computes a kernel that leads to a deterministic convergent value. This type of kernel is called a Neural Tangent Kernel (NTK). The NTK is random at initialization but varies during training as a Gradient Descent algorithm is used to make the data linearized in conjunction with a non-linearity Lipschitz function σ that we pre-apply to our data to allow our Gradient Descent to function the way we need it to. Using these kernel methods allows us to study and view ANNs from a different lens than how we're used to, and more specifically, study how kernels can help define ANNs, specifically positive-definite NTKs. Our theorem of focus, which we review and analyze, comes from the paper "Neural Tangent Kernel: Convergence and Generalization in Neural Networks" written by Arthur Jacot, Franck Gabriel and Clément Hongler, in particular the theorem (and proposition that precedes it) that shows how ANNs at an infinite-width limit and a non-linearity Lipschitz function can be equivalent to a GP, and then how we can find a deterministic convergence to our NTK as a result. We explore and analyze the implications of the values in the theorem as well as how the different components influence the limitations of what the theorem is applicable to. We discuss how one can verify the implications of the theorem through an empirical study that details what is implied by different results. Finally we propose the different directions for which future study may focus thanks to the implications brought about by our theorem of interest.

1 Introduction

Our report is a review of the paper Neural Tangent Kernel: Convergence and Generalization in Neural Networks by Arthur Jacot, Franck Gabriel, and Clément Hongler. With the wide spread prevalence of neural networks in use today it is well known that Artificial Neural Networks converge to a reasonable approximation of any function, provided that it is implemented with enough hidden neurons. Despite this, we are still uncertain as to what the optimization of ANNs converges to. And so, naturally this problem extends to deep neural networks where understanding the resulting convergence is of great interest. Understanding this will allow furthering the mathematical understanding of Neural Networks. In turn a better mathematical understanding allows further development and improvement of the ability to create models and allow us to begin to further describe other parts of ANNs. The mentioned paper seeks to provide an explicit result for the convergence of deep networks. Determining such a result has been a difficult task in the past since a large part of what neural networks do act as a black box and so we have been attempting to clear this up by bringing to light and describing these pieces in a more rigorous mathematical manner. To go into a detailed overview and review of this topic, we must first have an understanding of three major topics discussed in this report; Neural Networks, Gaussian Processes, and Kernels.

2 Convergence in Probability of the Neural Tangent Kernel to a Deterministic Limiting Kernel

We will first begin by stating our theorem of interest. Under the assumptions that we are considering fully-connected ANNs numbered from the input layer 0 to the output layer L each containing $n_i, i \in 1, ..., L$ neurons (the width of the layers) as well as the ANN having a Lipschitz, twice differentiable nonlinearity function $\sigma : \mathbb{R} \to \mathbb{R}$ with bounded second derivative. We take the primary theorem of the paper of Neural Tangent Kernel: Convergece and Generalization in Neural Networks by Arthur Jacot, Franck Gabriel, Clément Hongler for observation, it states.

Theorem 1. *For a network of depth L at initialization, with a Lipshitz nonlinearity σ, and in the limit as the layers width $n_1, ..., n_{L-1} \to \infty$, the NTK $\Theta^{(L)}$ converges in probability to a deterministic limiting kernel:*

$$\Theta^{(L)} \to \Theta_\infty^{(L)} \otimes Id_{nL}$$

Where the scalar kernel $\Theta_\infty^{(L)} : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}$ is defined recursively by

$$\Theta_\infty^{(1)}(x, x') = \Sigma^{(1)}(x, x')$$

$$\Theta_\infty^{(L+1)}(x, x') = \Theta_\infty^{(L)}(x, x')\dot{\Sigma}^{L+1}(x, x') + \Sigma^{(L+1)}(x, x'),$$

Where

$$\dot{\Sigma}^{L+1}(x, x') = \mathbb{E}_{f\sim\mathcal{N}(0,\Sigma^{(L)})}[\dot{\sigma}(f(x))\dot{\sigma}(f(x'))],$$

taking the expectation with respect to a centered Gaussian process f of covariance $\Sigma^{(L)}$, and where $\dot{\sigma}$ denotes the derivative of σ

Remark. *By Rademacher's theorem $\dot{\sigma}$ is defined everywhere, except perhaps on a set of zero Lebesgue measure*

This theorem helps with the understanding of ANNs in multiple ways. The first of which is it gives us a way to look at ANNs in another perspective, in this case we can look at them through Neural Tangent Kernels to describe the local dynamics of the ANN when preforming gradient descent. On top of this as a new link between Artificial Neural Networks and kernel methods it enables us to begin to describe ANNs in function space as opposed to paramater space.

3 Neural Networks

Now, let us discuss what a Neural Network is. Neural Networks as discussed in class, is a type of regression algorithm, specifically a regression tree algorithm. A Neural Network is essentially a set of prediction functions called Neural Computation Units.

Definition 3.1. (Neural Computation Unit).
A neural computation unit refers to a function with the form

$$f(\boldsymbol{X}) = \sigma(\beta^T \boldsymbol{X} + \beta_0)$$

where σ is some function,

$$\sigma(t) = \begin{cases} +1, & t \geq 0; \\ -1, & \text{otherwise.} \end{cases}$$

Definition 3.2. (Neural Network Model/Hypotheis).
A neural network refers to a family of prediction functions,

$F := \{f : \boldsymbol{X} \longrightarrow R; f$ can be represented by connecting different neural computation units.$\}$

Note that a Neural Network with a finite number of Neural Computation Units corresponds to a parametric model (This implies finite depth).

Now, the amount of layers of the recursive composition in a Neural Network model is called its depth, and a Neural Network with depth greater than or equal to 3 is called a Deep Neural Network. Furthermore, the width of a neural network is defined to be the maximal number of nodes in a layer. What we will be focusing on is the width of our Neural Networks, and the connection between an infinite width limit on Neural Networks that essentially makes them equivalent to Gaussian Processes.

4 Gaussian Processes

To define a Gaussian Process, we must first understand what it means to be Gaussian. A Gaussian random variable $X \sim N(\mu, \Sigma)$ where μ is the mean and Σ is the covariance matrix, with the following probability density function,

$$P(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|} e^{-\frac{1}{2}((x-\mu)^T \Sigma^{-1}(x-\mu))}$$

Now consider a regression problem(s):

$$y = f(x) + \epsilon \qquad \text{(Basic Regression Model)}$$
$$y = w^T x + \epsilon \qquad \text{(OLS and Ridge Regression)}$$
$$y = w^T \phi(x) + \epsilon \qquad \text{(Kernel Ridge Regression)}$$

The goal of regression is to estimate probabilities from our data. However, the above models give us a predictive model for our parameter $\boldsymbol{w}$. If we want to model the entire distribution of data, this is where the use of Gaussian Processes comes in.

Definition 4.1. (Gaussian Processes).

A Gaussian Process is a (potentially infinite) collection of random variables (RV) such that the joint distribution of every finite subset of RVs is multivariate Gaussian:

$$f \sim GP(\mu, k),$$

where $\mu(x)$ and $k(x, x')$ are the mean and co-variance functions respectively (Note that k is the kernel function of choice that generates our GP's co-variance matrix). Now, in order to model the predictive distribution $P(f_*|x_*, D)$ we can use a Bayesian approach by using a GP prior,

$$P(f|x) \sim N(\mu, \Sigma)$$

and condition it on the training data D to model the joint distribution of $f = f(X)$ (vector of training observations) and $f_* = f(x)$ (prediction at test input).

We know that Gaussian Processes can be used to do Regression, but first we need to fully understand what Kernels are, and how they generate our co-variance matrix Σ for our GPs.

Definition 4.2. (Kernels).

Intuitively, the Kernel is used as a measure of similarity. In particular, the Kernel Function $k(x, .)$ defines the distribution of similarities of points around a given point x. $k(x, y)$ denotes the similarity of a point x with another given point y.

What a Kernel does in actuality is a way of computing the dot product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in some (possibly very high dimensional) feature space, which is why Kernel Functions are sometimes denoted as "generalized dot products".

Suppose we have a mapping $\phi : R^n \longrightarrow R^m$ that brings our vectors in R^n to some feature space R^m. Then the dot product of $\boldsymbol{x}$ and $\boldsymbol{y}$ in this space is $\phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$. This is useful to us because it allows us to calculate dot products in some feature space without properly knowing how said feature space is defined.

With this basic understanding of Kernels and how they can help us compute our co-variance matrix Σ for our Gaussian Processes, we can see how their ability for high dimensionality calculation can give us a way to predict the co-variance for the whole distribution of our data.

Definition 4.3. Gaussian Process Regression (GPR).

First we will assume that before we observe the training labels, the labels are drawn from the zero-mean prior Gaussian Distribution:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_t \end{bmatrix} \sim N(0, \Sigma) \tag{1}$$

Without loss of generality (w.l.o.g) zero-mean is always possible by subtracting the sample mean. All training and test labels are drawn from an $(n+m)$-dimensional Gaussian distribution, where n is the number of training points, m is the number of testing points.

Remark. *The real training labels, $y_1, \cdots, y_n$ are samples of $Y_1, \cdots, Y_n$*

Now, with our kernel function calculating co-variance matrix Σ we can decompose Σ as,

$$\Sigma = \begin{bmatrix} K, K_* \\ K_*^T, K_{**} \end{bmatrix}$$

where K is the training kernel matrix, K_* is the training-testing kernel matrix, K_*^T is the testing-training kernel matrix and K_{**} is the testing kernel matrix. The conditional distribution of (noise free) values of the latent function $\boldsymbol{f}$ can be written as,

$$f_*|(Y_1 = y_1, Y_2 = y_2, \cdots, Y_n = y_n, x_1, x_2, \cdots, x_n, x_t) \sim N(K_*^T K^{-1} y, K_{**} - K_*^T K^{-1} K_*)$$

,

where the kernel matrices K_*, K_{**}, K are functions of $x_1, \cdots, x_n, x_*$

Now that we have clear definitions for Neural Networks and Gaussian Processes, we need to understand what a Neural Tangent Kernel so we can finally prove our Theorem.

5 Neural Tangent Kernel (NTK)

The main theorem we are reviewing from our paper revolves around our Neural Tangent Kernel converging in probability to a deterministic limiting kernel. This unique Neural Tangent Kernel is what we will hereby focus on understanding in the rest of our paper and how given the conditions in our theorem, it converges. First, allow me to give a brief example of a Neural Network and what the form of the NTK would be.

Assume we have a Neural network of Depth $L = 4$ and with a Lipschitz non-linearity σ. Let each parameter of the network, $w^{(0)}, w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)}$ belong to a vector $\theta = w^{(0)}, w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)}$ Each $w^{(i)}$ is initialized randomly and trained with Gradient Descent. Then we will have our Neural Network $f_\theta : R^{n_0} \longrightarrow R^{n_4}$. What our gradient descent does is make our inputs move in a bell-curve shape at each data point, creating a very Gaussian-like curve. See Figure 1.

What we are interested in specifically however, is the difference between the green curve above, and our data input x_0. This difference is what our NTK is, what it calculates. This results in the x_0 curve to become straight and smoothed out, and the same is said for every other value of x_0 at all the different initialization we may have. Furthermore, as we approach our infinite width limit, the tightness of our data points grows and grows until we achieve something similar to the following. See Figure 2.

Finally, if we add in another data point x_1, this will result in their relationship becoming linear. This relationship between two data-points is what we describe as our NTK,

$$\Theta^{(L)}_{(x_0,x_1)} = \sum_{w=1}^{W} \frac{d}{d\theta_w} f_\theta(x_0) \frac{d}{d\theta_w} f_\theta(x_1)$$

Where L is the depth of our Neural Network, x_0, x_1 are our two samples, and w is all of our parameters.

The main result we will focus on and review is how and why our NTK converges to a deterministic and fixed result as our hidden layers go to infinity (infinite width limit). See Figure 3.

6 Examples and Counterexamples

To start, let us continue with the example we used above to explain and define our NTK. Assume we have a Neural Network of Depth $L = 4$, with a Lipschitz non-linearity σ. Let each parameter of our ANN belong to $\theta = w^{(0)}, w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)}$. Each $w^{(i)}$ is initialized at random and trained with Gradient Descent. Thus we will have our ANN $f_\theta : R^{n_0} \longrightarrow R^{n_4}$. Recall that our NTK is $\Theta^{(L)}_{(x_0,x_1)} = \sum_{w=1}^{W} \frac{d}{d\theta_w} f_\theta(x_0) \frac{d}{d\theta_w} f_\theta(x_1)$. Our theorem discusses that in this situation, as the width of our ANN reaches infinity, $\Theta^{(L)}_{(x_0,x_1)}$ converges to a deterministic result. Since we define our ANN to have an infinite-width limit and a Lipschitz non-linearity σ, we know that our ANN's outputs tend to iid Gaussian processes.

Remark. *For a network of depth L at initialization, with a Lipschitz non-linearity σ, and in the limit as $n_1, \cdots, n_{L-1} \longrightarrow \infty$, the output functions f_θ, k for $k = 1, \cdots, n_L$ tend (in law) to iid centered Gaussian processes of co-variance $\Sigma^{(L)}$, where $\Sigma^{(L)}$ is defined recursively by:*

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0} x^T x' + \beta^2$$
$$\Sigma^{(L+1)}(x, x') = E_{f \sim N(0, \Sigma^{(L)})}[\sigma(f(x))\sigma(f(x'))] + \beta^2$$

taking the expectation with respect to a centered Gaussian process f of co-variance $\Sigma^{(L)}$

Thus, again referencing Figure 1, our NTK judges the distance between our ANN's output, and the Kernel Gradient result. This result is equivalent to the co-variance of our Gaussian Process, and reiterates why it is so important that we maintain an infinite-width limit for our ANN. Without the infinite-width limit, we would not have enough random initializations to be equivalent to a Gaussian Process, and have an accurate and deterministic co-variance value.
Let us analyze the proof for why a ANN with infinite-width limit has output functions that tend (in law of large numbers) to iid centered Gaussian processes of co-variance $\Sigma^{(L)}$

Proof. They prove this result using induction, starting with a ANN of depth $L = 1$; there are no hidden layers and f_θ is a random affine function of the form:

$$f_\theta(x) = \frac{1}{\sqrt{n_0}} W^{(0)} x + \beta b^{(0)}$$

All output functions $f_{\theta,k}$ are hence independent and have co-variance $\Sigma^{(1)}$ as needed.

The key to their induction step is that they assign $x = \alpha^{(L)}$, and do element-wise application of our Lipschitz non-linearity σ to these $\alpha^{(L)}$ values. By applying σ to, essentially our values x, we transform them into dependent, non-linear values that make use of our (required) fully-connected ANN. Furthermore, the definition of our Lipschitz non-linearity σ function results in a bound on our values of $\Sigma^{(L)}$, resulting in our co-variance to converge in probability to the expectation $E_{f \sim N(0, \Sigma^{(L)})}$ that is iid centered Gaussian with a deterministic, real co-variance of $\Sigma^{(L)}$

From here the induction step of $n_1, \cdots, n_{L-1} \longrightarrow \infty$ results in our $\alpha_i^{(L)}$ to tend to iid Gaussian processes with co-variance $\Sigma^{(L)}$

□

It is easy to see now that if we did not have our Lipschitz non-linearity function, or our ANN did not adhere to an infintie-width limit so that we could use the Law of Large Numbers, we would not be able to arrive at a deterministic Gaussian co-variance.

7 Empirical Studies

Based on the theorem of interest, an empirical study should confirm the convergence of the NTK as the width of the layers approaches infinity. We attempted to produce a study in R however continually ran into bugs with our code. So, to provide an outline of our process we will be outlining our thoughts and process here theoretically. In order to carry out such a study we must first have a dataset for which we can use as our training data. The most simple way that this data should be structured is such that there is a binary output for prediction on with any number of inputs preferably a smaller such that the prediction variables are less complex. Following this we should create multiple ANNs with the same number of hidden layers (each hidden layer within the same ANN should have equal widths) but different widths between each of the ANNs. For example we could create 3 seperate ANNs with 4 layers total (input, 2 hidden layers, binary output layer) where the widths of the hidden layers are 100, 1000, and 10000 for the respective ANNs. On these initialized ANNs we then need to calculate the initial NTK as stated within the theorem. Then we train the ANNs on the dataset and from the resulting trained ANNs calculate the NTK again as stated through the Theorem. These resulting outputs (NTK at initialization and NTK after training) should then be plotted onto a graph. The resulting differences between the NTKs at initialization and after training and the difference between each of the individual models with different layer widths should indicate a convergence towards some value thus confirming the theorem. Should the differences become larger or be sporadic this would have implied the opposite that the NTK was not converging.

8 Limitations of the Theorem

To begin looking at the limitations we must first identify what the required assumptions are. The limitations in this case are generally focused on the properties of the Artificial Neural Network. The assumptions we make are as follows:

1. The Artificial Neural Network is fully-connected
2. ANNs layers are labeled from 0 to L for input to output
3. Each layer contains n_i for $i \in 1, ..., L$ neurons
4. ANN has a Lipschitz nonlinearity function σ
5. the above nonlinearity function is $\mathbb{R} \to \mathbb{R}$

We can first observe simply that assumptions 2 and 3 do not bring us any limitations for the theorem. These are simply methods that we can specify as a way to label different components of the theorem. In any case that the labelings were different we could formulate a method to write it in these terms. In this case we would then have fixed the apparent "problem".

Assumption number 1, that the Artificial Neural Network is fully-connected is another such assumption that can be easily remedied. To begin with, an ANN is considered fully-connected if every neuron in layer i is connected to every neuron in layer $i+1$ $\forall$ $i \in 0, ..., L-1$. Therefore in the cases where an ANN is not fully connected we can stipulate that $\forall$ neurons in layer i such that $\exists$ a neuron

in layer $i+1$ that it is not connected to, add a connection with a fixed weight of 0. And so this in turn converts our ANN into a fully-connected ANN.

We now look at the more interesting assumptions, those being the remaining two assumptions, 4 which is the main part and 5. First, we start with looking at 4 since assumption 5 builds on this with specificity. In order to further determine how this influences the limitations of the Theorem we must first discuss what the nonlinearity function is for in the terms of the ANN. The nonlinearity function is a function σ which converts the input data for a single neuron into the output data that is fed into each neuron of the next layer. This in turn brings us to the first true limitation. If this were a linear function instead of a nonlinearity function. Then the ANN would begin to break down. In this case it would begin to act as if we were optimizing a linear function, and thus would be no different than optimizing a single layered neural network, since it would become one input tied to one output with intermediate layers being irrelevant. So, by limiting to a nonlinearity function we ensure that multiple layers are contribution to the final result. Now we arrive at the Lipschitz part of assumption 4. Here we observe the definition of Lipschitz.

Definition 8.1. Lipschitz (4)
Let $\mathrm{C} \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is ρ-Lipschitz over C if for every $w_1, w_2 \in \mathrm{C}$ we have that $\| f(w_1) - f(w_2) \| \leq \rho \| w_1 - w_2 \|$

So, from this definition of Lipschitz we can see that it entails that the function, in our case σ, is bounded from growing too quickly. This is an important restraint for applicable ANNs because if we do not have Lipschitz then we can not guarantee convergence of $\dot{\Sigma}^{L+1}(x, x')$ within the theorem. Finally we arrive at assummption 5, that $\sigma : \mathbb{R} \to \mathbb{R}$, this follows from the definition of Lipschitz. Here we are restriction the dimensions of the mapping. In theory this could be expanded to $\sigma : \mathbb{R}^d \to \mathbb{R}^k$ but this would be for different dimensionality of the nodes in the ANN. Since the nodes are singular we restrict to $\sigma : \mathbb{R}^1 \to \mathbb{R}^1$. And so, this limitation is more a limitation on the structure of the ANN than the theorem.

9 Future Directions

Now that we have taken a look at the different components of the theorem of interest as well as the assumptions that we make for the application of the theorem, we now turn to look at what future directions there are. With any theorem there opens up the idea of new areas of study or questions that only become visible once we have the theorem. Since our theorem of interest is related to the width of the networks layers there are always the obvious directions of looking further at other areas of uncertainty within neural networks, perhaps there exists an unknown property of ANNs that this theorem is the key that will unlock the ability to make progress on again. However from a more realistic point of view, with this theorem of interest being related to neural networks and convergence of properties that describe them, we are immediately met with two obvious questions of interest that can provide a more immediate direction of to look into.

The first of these questions is that of speed. Knowing that the NTK converges to some deterministic value is great, but we now are poised to ask, how fast does this convergence occur? The resulting answer to this is of great importance because it can have a profound impact on how long it takes to run an algorithm reliant on this theorem to completeness. In turn, the answer to this can have a large impact on what situations the application of this theorem is viable to use in, but it is a topic that first needs looking into before we can make such decisions.

The second question now follows and is then one of optimization. Though this may be related to speed in some ways there is still the question, how does knowing this help us to optimize neural networks. Or even, can it help us optimize them? To emphasize how this is distinct from the first question, this is a question of what is the best way to do something whereas the first question is how fast will this work. In short, this is more geared towards finding how we can speed it up or require

the least computation. For example could there be some formulations of nonlinearity functions σ that are more efficient than others?

10 Conclusion

Overall this theorem provides a new lens to view ANNs through. It provides new ways to think about ANNs by describing them in a manner related to other mathematical and statistical concepts. We believe that it will help to open up several future directions of research into ANNs as well as help with this progress. As requested in the final project sign up sheet, in the case of any bonus marks being provided we will be splitting them evenly, fifty-fifty.

References

[1] Neural Tangent Kernel: Convergence and Generalization in Neural Networks by Arthur Jacot, Franck Gabriel, and Clément Hongler,
`https://arxiv.org/pdf/1806.07572.pdf`

[2] DEEP NEURAL NETWORKS AS GAUSSIAN PROCESSES by Jaehoon Lee, Yasaman Bahri, Roman Novak , Samuel S. Schoenholz, Jeffrey Pennington, Jascha Sohl-Dickstein
`https://arxiv.org/pdf/1711.00165.pdf`

[3] Neural Tangent Kernel: Convergence and Generalization in Neural Networks - Video Example by Arthur Jacot,
`https://www.youtube.com/watch?v=raT2ECrvbag`

[4] Understanding Machine Learning: From Theory to Algorithms by Shai Shalev-Shwartz and Shai Ben-David

Appendix

There are two main proofs that are used for our results in this paper, derived from Reference [1], Neural Tangent Kernel: Convergence and Generalization in Neural Networks. The first proof details how we can equate an infinite-width limit ANN to be equivalent to iid centered Gaussian processes, and the second shows how we achieve the deterministic convergence on our NTK, and thus the result of our theorem in question.
Proof of Remark in 5 Examples and Counterexamples,

Remark. *For a network of depth L at initialization, with a Lipschitz non-linearity σ, and in the limit as $n_1, \cdots, n_{L-1} \longrightarrow \infty$, the output functions f_θ, k for $k = 1, \cdots, n_L$ tend (in law) to iid centered Gaussian processes of co-variance $\Sigma^{(L)}$, where $\Sigma^{(L)}$ is defined recursively by:*

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0} x^T x' + \beta^2$$
$$\Sigma^{(L+1)}(x, x') = E_{f \sim N(0, \Sigma^{(L)})}[\sigma(f(x))\sigma(f(x'))] + \beta^2$$

taking the expectation with respect to a centered Gaussian process f of co-variance $\Sigma^{(L)}$

Proof. We prove the result by induction. When $L = 1$, there are no hidden layers and f_θ is a random affine function of the form:

$$f_\theta(x) = \frac{1}{\sqrt{n_0}} W^{(0)} x + \beta b^{(0)}$$

All output functions $f_{\theta,k}$ are hence independent and have co-variance $\Sigma^{(L)}$ as needed.

They key to the induction step is to consider an $(L+1)$-network as the following composition: an L-network $R^{n_0} \longrightarrow R^{n_L}$ mapping the input to the pre-activations $\widetilde{\alpha}_i^{(L)}$, followed by an element-wise application of the non-linearity σ and then a random affine map $R^{n_L} \longrightarrow R^{n_{L+1}}$. The induction hypothesis gives that in the limit as sequentially $n_1, \cdots, n_{L-1} \longrightarrow \infty$ the pre-activations $\widetilde{\alpha}_i^{(L)}$ tend to iid Gaussian processes with co-variance $\Sigma^{(L)}$. The outputs

$$f_{\theta,i} = \frac{1}{\sqrt{n_L}} W_i^{(L)} \alpha^{(L)} + \beta b_i^{(L)}$$

conditioned on the values of $\alpha^{(L)}$ are iid centered Gauusians with co-variance

$$\widetilde{\Sigma}^{(L+1)}(x, x') = \frac{1}{\sqrt{n_L}} \alpha^{(L)}(x;\theta)^T \alpha^{(L)}(x';\theta) + \beta^2$$

.

By the law of large numbers, as $n_L \longrightarrow \infty$, this co-variance tends in probability to the expectation

$$\widetilde{\Sigma}^{(L+1)}(x, x') \longrightarrow \Sigma^{(L+1)}(x, x') = E_{f \sim N(0, \Sigma^{(L)})}[\sigma(f(x))\sigma(f(x'))] + \beta^2$$

. □

Proof for Theorem 1

Proof. This proof is again by induction. When $L = 1$, there is no hidden layer and therefore no limit to be taken. The NTK is a sum over the entries of $W^{(0)}$ and those of $b^{(0)}$:

$$\begin{aligned}\Theta_{kk'}(x, x') &= \frac{1}{n_0} \sum_{i=1}^{n_0} \sum_{j=1}^{n_1} x_i x_i^{'} \delta_{jk} \delta_{jk'} + \beta^2 \sum_{j=1}^{n_1} \delta_{jk} \delta_{jk'} \\ &= \frac{1}{n_0} x^T x' \delta_{kk'} + \beta^2 \delta_{kk'} = \Sigma^{(1)}(x, x') \delta_{kk'}.\end{aligned}$$

Here again, the key to prove this induction step is the observation that a network of depth $L+1$ is an L-network mapping the inputs x to the pre-activations of the L-th layer $\widetilde{\alpha}^{(L)}(x)$ followed be a non-linearity and a random affine function. For a network of depth $L+1$, let us therefore split the parameters in to the parameters of $\widetilde{\theta}$ of the first L layers and those of the last layer $(W^{(L)}, b^{(L)})$.

By the Remark above, and the induction hypothesis, as $n_1, \cdots, n_{L-1} \longrightarrow \infty$ the pre-activations $\widetilde{\alpha}_i^{(L)}$ are iid centered Gaussians with co-variance $\Sigma^{(L)}$ and the NTK $\Theta_{ii'}^{(L)}(x, x')$ of the smaller network converges to a deterministic limit:

$$\left(\partial_{\widetilde{\theta}} \widetilde{\alpha}_i^{(L)}(x;\theta)\right)^T \partial_{\widetilde{\theta}} \widetilde{\alpha}_{i'}^{(L)}(x';\theta) \longrightarrow \Theta_{\infty}^{(L)}(x, x') \delta_{ii'}$$

We can split the NTK into a sum over the parameters $\widetilde{\theta}$ of the first L layers and the remaining parameters $W^{(L)}$ and $b^{(L)}$.

For the first sum let us observe that by the chain rule:

$$\partial_{\widetilde{\theta}_p} f_{\theta,k}(x) = \frac{1}{n_L} \sum_{i=1}^{n_L} \partial_{\widetilde{\theta}_p} \widetilde{\alpha}_i^{(L)}(x;\theta) \dot{\sigma}(\widetilde{\alpha}_i^{(L)}(x;\theta)) W_{ik}^{(L)}$$

.

By the induction hypothesis, the contribution of the parameters $\widetilde{\theta}$ to the NTK $\Theta_{kk'}^{(L+1)}(x, x')$ therefore converges as $n_1, \cdots, n_{L-1} \longrightarrow \infty$:

$$\begin{aligned}&\frac{1}{n_L} \sum_{i,i'=1}^{n_L} \Theta_{ii'}^{(L)}(x, x') \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_{i'}^{(L)}(x';\theta)\right) W_{ik}^{(L)} W_{i'k'}^{(L)} \\ \longrightarrow &\frac{1}{n_L} \sum_{i,=1}^{n_L} \Theta_{\infty}^{(L)}(x, x') \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x;\theta)\right) \dot{\sigma}\left(\widetilde{\alpha}_i^{(L)}(x';\theta)\right) W_{ik}^{(L)} W_{ik'}^{(L)}\end{aligned}$$

By the law of large numbers, as $n_L \longrightarrow \infty$, this tends to its expectation which is equal to

$$\Theta_{\infty}^{(L)}(x, x')\dot{\Sigma}^{(L+1)}(x, x')\delta_{kk'}$$

It is then easy to see that the second part of the NTK, the sum over $W^{(L)}$ and $b^{(L)}$ converges to $\Sigma^{(L+1)}\delta_{kk'}$ as $n_1, \cdots, n_L \longrightarrow \infty$.

□

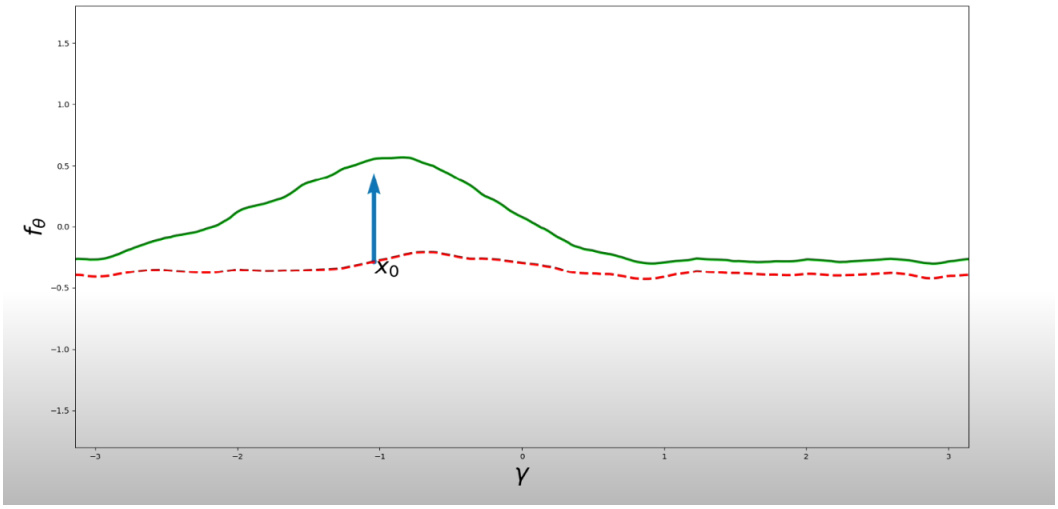

Figure 1: Gradient Descent on a point randomly initialized point x_0 from our Neural Network

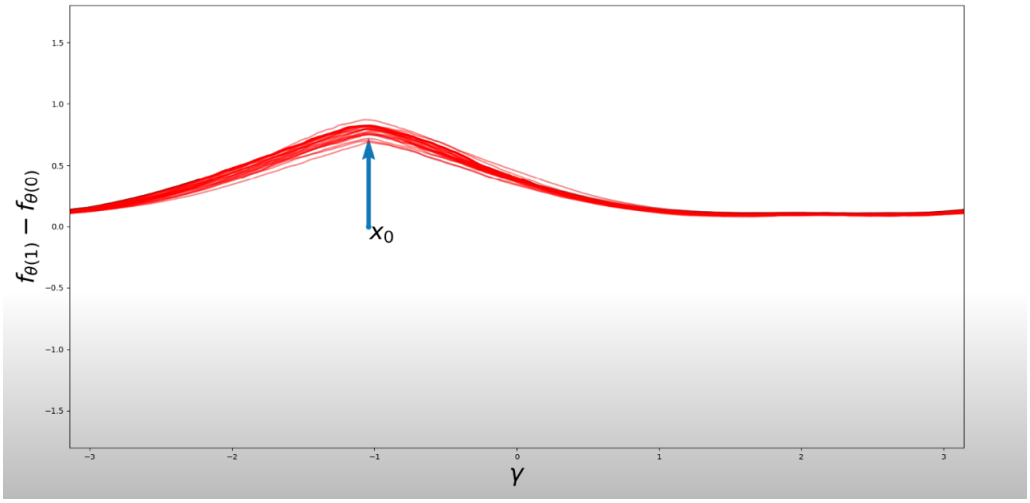

Figure 2: Neural Tangent Kernel for the point x_0 at all initializations (each line is another hidden layer of our Neural Network).

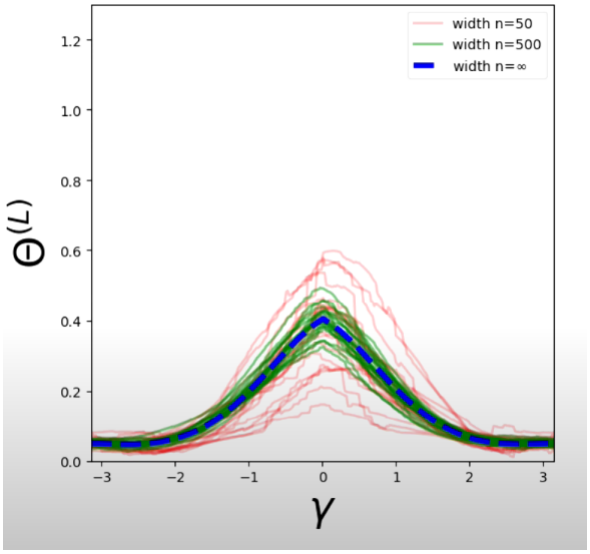

Figure 3: Neural Tangent Kernel converging as Neural Network width increases to infinity